INFINITE IMPULSE RESPONSE MULTIPLIERLESS DIGITAL FILTER ARCHITECTURE

Technical Field

[0001] The present invention relates to digital signal processing, and, more specifically, to IIR digital filter designs avoiding the use of multipliers and implemented in a VLSI architecture.

Background of the Invention

[0002] Digital signal processing (DSP) provides tremendous benefits in manipulating analog information, such as audio, still pictures, and video, that have been converted into digital information. DSP improves the accuracy and reliability of digital communications and is used extensively in numerous applications throughout modern society. DSP operates by clarifying and standardizing the level or states of a digital signal. In so doing, a DSP circuit is able to differentiate between human-made signals, which are ordered, and noise, which is inherently chaotic.

[0003] Most digital signal processing functions require digital filters. Digital filters are used for many purposes including reducing noise from corrupted signals, transforming signals in the frequency domain, and changing certain characteristics of a signal to a desired characteristic. Specific applications of digital filters include noise and distortion reduction in cellular phones, multipath cancellation in digital television systems, reducing distortion in digital modems, filtering operations in CD

players, GPS systems, and target detection and other forms of military hardware. As can be appreciated, numerous other applications exist in the ever-expanding product lines of consumer electronics.

[0004] The consumer electronic industry is always seeking improved algorithms for filter designs. Areas of improvement include faster filtering, lower power consumption, and less chip area. Faster filtering improves processing speeds and lower power translates into less expensive operating costs. Reducing the chip area means more functions can be incorporated into the same chip which is the trend of today's consumer electronics. Less chip area also reduces the overall cost of the chip.

[0005] Thus, it would be an advancement in the art to provide a filter design with increased speed, lower power, and reduced chip area. Such an invention is disclosed and claimed herein.

Summary of the Invention

[0006] The present invention provides an improved Infinite Impulse Response (IIR) filter with feedback and forward coefficients. For VLSI implementation, multiplierless filters are faster, more compact, and use less power than filters with multipliers. Multiplication of any number with a power-of-two number is a shift operation. Through shift operations, a filtering technique eliminates multipliers and their disadvantages.

[0007] To increase filtering performance, power-of-two coefficients of the designed filter are derived using a genetic algorithm. Genetic algorithms are based on a biological metaphor that views the process as a competition among a population of evolving candidate problem solutions. A fitness function evaluates each solution to

decide whether it will contribute to the next generation of solutions. Then, through operations analogous to gene transfer in sexual reproduction, the algorithm creates a new population of candidate solutions. The process continues until the solution meets the target value or the genetic algorithm does not yield better results after a certain number of iterations.

[0008] The IIR filter design of the present invention includes a memory with the power-of-two coefficients generated by a genetic algorithm stored thereon. The filter further includes two shift registers to receive input samples and previous outputs. A shifter stage is coupled to the memory and the two shift registers to receive the power-of-two coefficients, input samples, and previous outputs. The shifter stage performs shift operations for input samples and previous outputs based on corresponding power-of-two coefficients.

[0009] The products generated by the shifter stage are transmitted to an adder stage that adds the products in parallel. The filter architecture incorporates sequential pipelining for computing the output at a very fast rate. The sums are then accumulated to produce an outputted estimate.

[0010] The filter architecture provides a smaller, faster, and cheaper digital filter than conventional designs. Additional aspects and advantages of this invention will be apparent from the following detailed description of preferred embodiments, which proceeds with reference to the accompanying drawings.

Brief Description of the Drawings

[0011] Non-exhaustive embodiments of the invention are described with reference to the figures in which:

[0012] Figure 1 is a block diagram of a filter design architecture;

[0013] Figure 2 is a block diagram of a filter design architecture; and

[0014] Figure 3 is a block diagram of an adder tree structure.

Detailed Description of Preferred Embodiments

[0015] Reference is now made to the figures in which like reference numerals refer to like elements. For clarity, the first digit or digits of a reference numeral indicates the figure number in which the corresponding element is first used.

[0016] Throughout the specification, reference to "one embodiment" or "an embodiment" means that a particular described feature, structure, or characteristic is included in at least one embodiment of the present invention. Thus, appearances of the phrases "in one embodiment" or "in an embodiment" in various places throughout this specification are not necessarily all referring to the same embodiment.

[0017] Furthermore, the described features, structures, or characteristics may be combined in any suitable manner in one or more embodiments. Those skilled in the art will recognize that the invention can be practiced without one or more of the specific details, or with other methods, components, materials, etc. In other instances, well-known structures, materials, or operations are not shown or not described in detail to avoid obscuring aspects of the invention.

[0018] The present invention relies on evolutionary algorithms to provide coefficient values for implementation in the design architecture. The field of evolutionary algorithms includes problem solving systems that involve evolutionary processes as key elements in the design and implementation. A variety of evolutionary algorithms exist with some of the more well known being genetic algorithms, evolutionary programming, evolution strategies, classifier systems, and genetic programming.

[0019] All of the evolutionary algorithms share a common concept of simulating the evolution of individual structures via processes of selection, mutation, and reproduction. The processes depend on the perceived performance of the individual structures as defined by an environment.

[0020] Evolutionary algorithms maintain a population of structures, that evolve according to rules of selection and other operators, which are referred to as search operators. Each individual in the population receives a measure of its fitness in the environment. Reproduction focuses attention on high fitness individuals to exploit the available fitness information. Recombination and mutation perturb those individuals and provide general heuristics for exploration. The evolutionary algorithms are sufficiently complex to provide robust and powerful adaptive search mechanisms.

[0021] The present invention incorporates genetic algorithms to optimize the filter design coefficients. The genetic algorithm is a model of learning that derives its behavior from a metaphor of the processes of evolution in nature. This is done by the creation of a population of individuals represented by chromosomes, in essence a set of character strings. The individuals in the population then experience a process of evolution. The process of evolution requires individuals to compete for resources in an environment. Those individuals that are more successful are more likely to survive and propagate their genetic material.

[0022] In practice, the genetic model of computation is implemented in arrays of bits or characters to represent the chromosomes. The genetic algorithm is not a random search for a solution to the problem of selecting the highly fit individual. The

genetic algorithm uses a stochastic process, but the result is better than a random process.

[0023] Genetic algorithms are used for various applications. One example, would be multidimensional optimization problems in which the character string of the chromosome can be used to encode the value for the different parameters being optimized.

[0024] When a genetic algorithm is implemented, it is usually done in a manner that involves the following cycle: evaluate the fitness of all of the individuals in the population; create a new population by performing operations such as fitness-proportionate reproduction on the individuals whose fitness has just been measured; and discard the old population and iterate using the new population.

[0025] One iteration of this cycle is referred to as a generation. The first generation of this process operates on a population of randomly generated individuals. From there on, the genetic operations operate to improve the population.

[0026] The present invention uses a genetic algorithm to generate power-of-two coefficients for a one dimensional stable IIR filter. An IIR filter is a type of digital signal filter in which each sample of an output is the weighted sum of past and current samples of the input and past samples of the output. Thus, an IIR Filter produces an output, *y(n)*, that is the weighted sum of the current and past inputs, *x(n)*, and past outputs.

[0027] The output of a general IIR filter is given by:

$$y_n = \sum_{i=1}^{p} a_i y_{n-i} + \sum_{j=0}^{q} b_j x_{n-j},$$

[0028] where *p* is a non-zero value. Common types of IIR filters are Butterworth, Bessel, Chebyshev, and Elliptic.

[0029] The filter design is based on using the genetic algorithm to obtain the power-of-two coefficients such that the square error of their frequency responses is minimized. As a result, the designed filter, whose frequency response is closely matched to the frequency response of the desired filter, can be implemented using simple shifting operations without multipliers.

[0030] A desired IIR filter may be given as:

$$y(n) + a_1 y(n-1) + \ldots + a_m y(n-M) = b_0 x(n) + b_1 x(n-1) + \ldots + b_N x(n-N).$$

[0031] A genetic algorithm is employed to find the coefficients of the designed multiplierless filter which may be given as:

$$y(n) + A_1 y(n-1) + \ldots + A_p y(n-P) = B_0 x(n) + B_1 x(n-1) + \ldots + B_Q x(n-Q).$$

[0032] A genetic algorithm minimizes the cost function (squared error), $E = \sum (h(n) - h_{designed}(n))^2$, for all *n* in a region of support of *h(n)* where *h(n)* is the impulse response of the desired filter and $h_{designed}(n)$ is an impulse response of the designed multiplierless filter. The order of the multiplierless filter, *P*, can be greater than or equal to the order of the desired filter, *M*.

[0033] The genetic algorithm defines a chromosome or an array of parameter values to be optimized. In the present invention, the parameter values are all filter coefficients $\{A_1, \ldots, A_p, B_0, B_1, \ldots, B_Q\}$. An integer genetic algorithm is employed to obtain the result.

[0034] The chromosome elements are represented by an integer which represents the power-of-two coefficient. By way of example, the set,

$S\{n\} = \{\pm 2^{-8}, \pm 2^{-7}, \ldots \pm 2^{-1}, \pm 1\}$, represents a set of 17 integers accounting for the sign. One way to code the power-of-two coefficients is $\{-2^{-8}, -2^{-7}, \ldots, -2^{-1}, -1, 0, 1, \ldots, 2^{8}\} = \{0, 1, \ldots, 7, 8, 9, \ldots, 15, 16\}$. Each chromosome is associated with a cost value found by evaluating the cost function. N_{pop} is the total number of chromosomes and the genetic algorithm performs the following steps.

[0035] In an initial step, the genetic algorithm provides an initial population by generating $2N_{pop}$ chromosomes randomly. The genetic algorithm ranks the costs defined in the cost function from lowest cost to highest cost. The best N_{pop} chromosomes are kept for each iteration, while the others are discarded.

[0036] In a second step, the genetic algorithm selects and pairs N_{pop} chromosomes. Using a weighted random pairing scheme based on the cost of each chromosome, two chromosomes are randomly selected to produce two new chromosomes called offsprings. Pairing continues until new offsprings are generated to replace the discarded offspring. The population size N_{pop} is constant for all iterations.

[0037] In a third step, the genetic algorithm generates two new offspring from each pair of chromosomes in the second step by crossover operation with a probability P_c. The paired chromosome exchanges a part of themselves at the crossover point, which is randomly selected between the first and last bits of the paired chromosomes.

[0038] In a fourth step, the genetic algorithm keeps the best (lowest cost) chromosome and applies a mutation function to the rest of the population with probability P_m. The mutation function replaces a chromosome with a new integer in the same set S{n} at the mutation point which is also randomly selected.

[0039] The foregoing steps are repeated until the minimum solution meets the target value or the genetic algorithm does not yield better results after a certain number of iterations.

[0040] Referring to Figure 1, a block diagram is shown of an architecture for an IIR filter 10. The filter 10 includes a memory 12, such as a ROM or a variation thereof, that contains the power-of-two coefficients $\{a_1,\ldots a_M, b_0,\ldots b_N\}$ generated by a genetic algorithm. The present invention increases filter speed by limiting the design coefficients' to a power-of-two. Since the filter design is multiplierless, it is implemented with very small number of shifts and no multiplications. This allows for a very high throughput, consumes low power, since it has low gate count, and saves on chip area.

[0041] In one implementation, the memory 12 is a collection of single port ROMs for storing coefficients that are 9 bits wide. The number of words stored in the memory depends on the filter design constraints and the memory 12 may be configured to store any number of coefficients. By way of example, the memory 12 may be embodied as 32 independent ROMs, each of which is 8-words deep. The memory 12 may also be embodied in numerous other implementations based on application needs.

[0042] The filter 10 further includes a first shift register 18 that receives and stores input samples *x(n)*. In one common implementation, the first shift register 18 may be embodied as a 64-word deep, 9 bit wide shift register. When a new input sample *x(n)* is received, it is stored as the current sample and all previous samples are shifted by one storage location with the oldest sample discarded. As such, the first shift register 18 stores 64 input samples.

[0043] The filter 10 includes a second shift register 20 that receives previous outputs *y(n-1)*. The second shift register 20 may be embodied as a 192-word deep, 9 bit wide shift register. When a new output *y(n)* is created, it is stored in the second shift register 20 as the current output and all previous outputs are shifted by one storage location with the oldest output discarded. As such, the second shift register store 192 previous outputs.

[0044] As can be appreciated by one of skill in the art, the shift registers 18, 20 may be configured in various ways and still be within the scope of the present invention. In one embodiment, 64 input samples *x(n)* to *x(n-63)* and 192 previous outputs *y(n-1)* to *y(n-192)* are stored in the shift registers 18, 20 to accommodate conventional practice for ghost cancellation for HDTV. The present invention is applicable for filtering techniques requiring various numbers of input samples and previous outputs. Thus, references to specific examples are for illustrative purposes only and should not be considered limiting of the scope of the invention.

[0045] The first shift register 18 is in communication with a first multiplexer 22 to combine the input samples *x(n)*. In one implementation, the first multiplexer 22 is embodied as 8 8-input multiplexers. The second shift register 20 is in communication with a second multiplexer 24 to combine the previous outputs *y(n-1)*. In one implementation, the second multiplexer 24 is embodied as 24 8-input multiplexers.

[0046] The shift registers 18, 20 are both in communication with a shifter stage 26 via multiplexers 22, 24. The memory 12 is also in communication with the shifter stage 26. The shifter stage 26 receives the power-of-two coefficients from the memory 12, the input samples *x(n)* from the first multiplexer 22, and the previous

outputs *y(n-1)* from the second mutliplexer 24. The shifter stage 26 performs a shift operation to provide products. The shifter stage 26 shifts the input samples *x(n)* and previous outputs *y(n-1)* by the corresponding power-of-two coefficients $\{A_1,...A_M, B_0,...B_N\}$.

[0047] The memory 12, shift registers 18, 20, and the multiplexers 22, 24 are in communication with a controller 28 that controls their respective operation. The controller 28 addresses a zero memory location and all other locations in the memory 12 and the shift registers 18, 20 via the multiplexers 22, 24. Thus, when the controller 28 places the address zero, the first coefficient and the corresponding input sample are transmitted to the shifter stage 26. In the next clock cycle, the shifter stage 26 provides the product of the first coefficient and the corresponding input sample.

[0048] The shifter stage 26 provides all products to an adder stage 30 which sums the products. The adder stage 30 is in communication with an accumulator 32 that accumulates the products provided by the adder stage 30 and produces an output estimate, *y(n)*. After providing *y(n)* the accumulator 32 resets itself to zero to start accumulating for the new estimate. The resulting output *y(n)* is then transmitted to the shift register 20 for subsequent operations.

[0049] Referring to Figure 2, a block diagram of an IIR filter design 10 is shown in greater detail. The shifter stage 26 includes barrel shifters 34 that are disposed in parallel. Each barrel shifter 34 receives a power-of-two coefficient and a corresponding input sample or previous output. Each barrel shifter then provides a product of the power-of-two coefficient and the corresponding input sample or previous output. The number of barrel shifters 34 may vary based on design

constraints and application. An increased number of barrel shifters 34 increases speed but requires more chip space. In one implementation, 32 barrel shifters may be used to complete 256 products in 8 clock cycles.

[0050] The barrel shifters 34 provide outputted products to the adder stage 30 that includes one or more adder trees 36. In the illustrated example, the adder stage 30 includes two adder trees 36 disposed in parallel to sum the received products. As can be appreciated, more or fewer adder trees 36 may be used based on design constraints. Each adder tree 36 is in communication with an accumulator 32 which accumulates the output of the adder trees 36 and provides an estimate, $y(n)$. In the illustrated architecture, the accumulator 32 will provide an output, $y(n)$, after 8 clock cycles.

[0051] Referring to Figure 3, a block diagram of one embodiment of an adder tree structure 36 is shown. The adder tree 36 includes a series of adder elements 40 disposed in a series. Each adder element 40 includes a number of adders 42 disposed in parallel. In sequence, the adder elements 40 may include progressively fewer adders as the values are combined to a final sum. Furthermore, the adders 42 may sequentially progress in increased bit capacity as values increase.

[0052] In one example, an adder tree 36 receives 16 product inputs which are sent to a first adder element 40 having eight adders 42 to provide eight outputs. The 16 product inputs are generated by 16 corresponding barrel shifters. The adders 42 may be 17 bit to accommodate the inputs. The next adder element 40 has four adders 42, such as 18 bit adders, that receive eight inputs and provide four outputs. The subsequent adder element 40 has two adders 42, such as 19 bit adders, that

receive four inputs and provide two outputs. A final adder element 40 has one adder 42, such as a 20 bit adder, that receives two inputs and provides a final sum.

[0053] In this example, the adder tree 36 initially takes 4 clock cycles to produce a sum of 16 inputs, but since the adder tree 36 is in a pipeline configuration the adder tree 36 produces a sum of 16 inputs every clock cycle afterwards. The illustrated adder tree structure 36 is for exemplary purposes only and numerous variations may be made in the adder elements 40 and the adders 42 themselves without departing from the scope of the invention.

[0054] The IIR filter design of the present invention provides a smaller, faster, and cheaper digital filter than conventional designs. The filter design provides a low gate count and increased speed by using power-of-two coefficients. The filter design incorporates parallelism and sequential pipelining to accommodate limited resources in circuit design and to operate at a faster clock. In operation, it is beneficial to receive an estimate, $y(n)$, in 9 clock cycles. The present invention allows operation at a sampling time as fast as 9 times the clock period which is required for many applications.

[0055] It will be obvious to those having skill in the art that many changes may be made to the details of the above-described embodiments without departing from the underlying principles of the invention. The scope of the present invention should, therefore, be determined only by the following claims.